Defence Research and Development Canada
Recherche et développement pour la défense Canada

DEFENCE R&D DÉFENSE

Sigmoidal Weight Constraint in a Recurrent Neural Network

Simon A. Barton
Defence R&D Canada – Suffield

Technical Memorandum
DRDC Suffield TM 2004-261
December 2004

Canada

Sigmoidal Weight Constraint in a Recurrent Neural Network

Simon A. Barton

Defence R&D Canada – Suffield

Technical Memorandum

DRDC Suffield TM 2004-261

December 2004

106-03827/

Author

Simon A. Barton

Approved by

D.M. Hanna
Head/TVSS

Approved for release by

Dr Paul D'Agostino
Chair, Document Review Panel

© Her Majesty the Queen as represented by the Minister of National Defence, 2004

© Sa majesté la reine, représentée par le ministre de la Défense nationale, 2004

Abstract

When training a recurrently connected neural network (RNN), the magnitude of the connection strengths (weights) must be limited in some way. The weights are normally constrained by either renormalizing them after each learning step, or by using a decay term proportional to the weight. For large numbers of training cycles, we show that an RNN output can become unstable with previously used weight adjustment methods.

We introduce a technique that constrains weight values to move on a smooth sigmoidal curve. Without the need for renormalization or a parametric decay term, our RNNs then produce stable output. Performance is also improved in other ways. As an example, an associative memory RNN is shown to converge much faster and to more accurate values than with previous methods.

Résumé

En effectuant l'entraînement d'un réseau récurrent de neurones (RRN), la magnitude des forces (poids) de la connexion doit être limitée de quelque façon. Les poids sont normalement freinés soit par leur renormalisation après chaque étape d'apprentissage soit par l'utilisation d'un facteur de décroissance proportionnel au poids. On a montré, qu'après avoir subi une grande quantité de cycles d'apprentissage, une sortie RRN peut devenir instable à cause des méthodes d'ajustement des poids utilisées antérieurement.

On a introduit une technique qui freine les valeurs des poids pour les faire déplacer selon une courbe sigmoïdale douce. Notre RRN produit alors des sorties stables sans avoir besoin d'être renormalisé ou d'utiliser un facteur de décroissance. La performance est alors améliorée par d'autres moyens. Comme exemple, on montre ici qu'un RRN à mémoire associative peut converger plus rapidement et à des valeurs plus exactes qu'avec les méthodes précédentes.

This page intentionally left blank.

Executive summary

The development of autonomous land vehicles is an important part of R&D at DRDC Suffield.

Part of this research is expanding methods of artificial intelligence (AI) that may be applied to autonomous vehicle navigation, sensing and control. A recurrent neural network (RNN) is an AI technique that has several useful functions. Memory of sensor information, and the formation of relationships between patterns from different types of sensor is one such function. The use of sensor input to drive action is another. In previous work we used the output from RNNs to store and identify patterns, to generate associations between input patterns and identifications, and to direct the motion of a simulated mobile machine to follow a moving goal. The long term intention of this work is to use memory and pattern association to control and improve machine action to achieve goals.

The structure and learning mechanisms of an RNN are based on biological neural systems. An RNN is a collection of processing nodes with multiple feedback connections. The strength of the connection between two nodes is called the weight, and the output of an RNN is essentially determined by the values of these weights.

During training, the magnitudes of the weights are increased or decreased according to a learning rule. With simple weight adjustment schemes the magnitudes of the weights may grow indefinitely under some circumstances, and so their magnitudes must be limited in some way. The weights are normally constrained by either renormalizing them after each learning step, or by using a decay term proportional to the weight. When training is continued over thousands of cycles, we show that an RNN output can become unstable with previously used weight adjustment methods.

We introduce a technique that constrains weight values to move on a smooth sigmoidal curve. Without the need for renormalization or a parametric decay term, our RNNs then produce stable output. Performance is also improved in other ways. As an example, an associative memory RNN is shown to converge much faster and to more accurate values than with previous methods.

Computational time is thus greatly reduced and accuracy is significantly improved for any RNN application that uses a trained network. Continuous training may also be employed while maintaining numerical stability.

The sigmoidal technique has also been employed in an RNN controlling the movement of a simulated mobile machine. Since this employs continuous learning it benefits greatly from the stability provided by the sigmoidal weight constraint technique.

Part of this work was performed under a Technology Investment Fund project, entitled, "Self-Organized, Goal-Driven Adaptive Learning", and continues under project 12ph - Autonomous Land Systems.

Barton, Simon A. 2004. Sigmoidal Weight Constraint in a Recurrent Neural Network. DRDC Suffield TM 2004-261. Defence R&D Canada – Suffield.

Sommaire

À RDDC Suffield, la mise au point de véhicules terrestres autonomes représente une partie importante de la R & D.

Une partie de cette recherche consiste à étendre les méthodes d'intelligence artificielle (IA) pouvant être appliquées à la navigation, aux capteurs et au contrôle des véhicules autonomes. Un réseau récurrent de neurones est une technique IA qui possède plusieurs fonctions utiles. La mise en mémoire d'informations provenant des capteurs et la mise en rapport de formes de types différents de détecteurs sont l'une de ces fonctions. L'utilisation des entrées provenant des capteurs pour diriger les actions en est une autre. Lors des travaux précédents, nous avons utilisé les sorties provenant des réseaux récurrents de neurones pour mettre en mémoire et identifier les formes, pour générer des associations entre les formes d'entrée et des identifications ainsi que pour diriger le mouvement d'une machine mobile simulée poursuivant un objectif en motion. Le but à long terme de ces travaux et d'utiliser la mémoire et l'association des formes pour contrôler et améliorer les actions de la machine vers ses objectifs.

La structure et les mécanismes d'apprentissage d'un RRN sont basés sur les systèmes biologiques neuraux. Un RRN est une collection de nœuds de traitement ayant des connexions multiples de rétroaction. La force de la connexion entre deux nœuds est appelée le poids et la sortie d'un RRN est essentiellement déterminée par les valeurs de ces poids.

Durant l'entraînement, les magnitudes de ces poids ont été augmentées ou diminuées selon une règle d'apprentissage. Un simple ajustement du poids permettant aux schémas des magnitudes des poids de continuer à augmenter pendant une période indéterminée dans certaines circonstances, leurs magnitudes doivent être limitées par quelque moyen. Les poids sont normalement freinés soit en les renormalisant après chaque étape d'apprentissage soit en utilisant un facteur de décroissance proportionnelle au poids. Si on continue l'apprentissage pendant plusieurs milliers de cycles, on s'aperçoit qu'une sortie RRN peut devenir instable à cause des méthodes d'ajustement de poids antérieurement utilisées.

On a introduit une technique qui freine les valeurs des poids pour les faire déplacer selon une courbe sigmoïdale douce. Notre RRN produit alors des sorties stables sans avoir besoin d'être renormalisé ou d'avoir un facteur de décroissance. La performance est alors améliorée par d'autres moyens. Comme exemple, on montre ici qu'un RRN à mémoire associative peut converger plus rapidement et à des valeurs plus exactes qu'avec les méthodes précédentes.

La durée de calcul est ainsi grandement réduite et l'exactitude est améliorée de manière signifiante pour toutes les applications RRN qui utilisent un réseau entraîné. Il est possible d'employer un apprentissage continu tout en maintenant la stabilité numérique.

La technique sigmoïdale a aussi été employée par un RRN contrôlant le mouvement d'une machine mobile simulée. Ceci emploie un apprentissage continue et bénéficie grandement de la stabilité fournie par la technique sigmoïdale de freinage du poids.

Une partie de ces travaux a été effectuée grâce au projet du Fonds d'investissement en technologie appelé « Apprentissage adaptatif, auto-organisateur et centré sur les objectifs à atteindre » et continue sous le projets 12 ph – les systèmes terrestres autonomes.

Barton, Simon A. 2004. Sigmoidal Weight Constraint in a Recurrent Neural Network. DRDC Suffield TM 2004-261. R & D pour la défense Canada – Suffield.

Table of contents

Abstract i

Résumé i

Executive summary iii

Sommaire v

Table of contents vii

List of figures viii

1 Introduction 1

2 An Associative Memory RNN 3

2.1 Node Response Functions 4

2.2 Changing the Connection Weights 4

2.3 The Correlation Function 6

2.4 Sigmoidal Weight Constraint 6

3 Results 9

3.1 Long Term Instability with No Sigmoidal Weight Constraint 9

3.2 Stable, Rapid Convergence with Sigmoidal Weight Constraint 9

4 Discussion and Conclusions 13

References 14

List of figures

1 Sigmoidal curve used to constrain weights 7

2 Erratic correlations for images (lower) and codes (upper), using 500 R nodes to train 10 images ($\rho = 0.001$) 10

3 Unstable correlations for images (lower) and codes (upper), using 500 R nodes to train 10 images ($\rho = 0.002$) 10

4 Correlations for 10 images (lower) and codes (upper), using Sigmoidal Weight Constraint ($\rho = 0.001$) 11

5 Correlations for 10 images (lower) and codes (upper), using Sigmoidal Weight Constraint ($\rho = 0.002$) 12

1 Introduction

An artificial neural network is a collection of interconnected processors, called nodes. Each node receives input from many other nodes in the ensemble. The number of inputs at each node and the location of the connected nodes can be varied. The strength of each connection is represented by a floating point number, called the weight.

Each node sums the weighted input from all connected nodes and passes the result through a response function. Thus, the weights between the nodes determine the output of the network. The form of the response function used for each node is the only other factor determining the output. The weights are altered while the network is being trained to perform a particular function.

If there is feedback within the network, using recurrently connected nodes, the way in which the weights are altered during training and subsequent operation will strongly affect the internal dynamics, which can be convergent, periodic or chaotic [1].

In previous work, we have used the output from convergent recurrent neural networks (RNNs) to store and identify patterns [2],[3], to generate associations between input patterns and output codes [4], and to direct the motion of a simulated mobile machine to follow a moving goal [5],[6].

During training, the magnitudes of the weights connecting two nodes are increased or decreased according to a learning rule. The problem with simple weight adjustment schemes is that the magnitudes of the weights may grow indefinitely under some circumstances, e.g. using a sequence of repeated patterns with some regions that have a constantly strong output. The weight magnitudes must therefore be limited in some way. The weights are normally constrained by either rescaling (normalizing) them after each learning step, or by using a decay term proportional to the weight [7],[8].

One of our learning rules (a form of Hebbian learning [9]) used a decay term to constrain the weight magnitudes. This learning rule then contained both growth and decay rate parameters. This has two disadvantages: 1) the two parameters must be optimized; and 2) the competition between growth and decay may introduce numerical instability in finely balanced systems. For pattern storage applications we also used a form of difference feedback learning learning [4] that has no explicit weight constraint and can also introduce numerical instability, as we show in Chapter 3.

This work descibes a novel technique for constraining the weight magnitudes without the need for either explicit renormalization or the introduction of a parametric decay term.

In Chapter 2 we review the RNN structure and learning rules that we have previously used for associative pattern memory, and we introduce the new constraint technique, which confines the weight values to move on a smooth sigmoidal curve.

Chapter 3 presents results that show numerical instability using the learning algorithms with no sigmoidal weight constraint. We then show the effects on the RNN performance

and stability when the sigmoidal constraint is used. The number of network training cycles required to accurately store a set of patterns and the accuracy of the storage and identification are used as performance criteria.

The results presented in Chapters 3 use an associative memory to demonstrate RNN performance with our sigmoidal weight contraint technique. In addition, the technique has been successfully appled to RNNs that perform other tasks, specifically goal-directed motion of a simulated mobile machine, and relational learning using two input image streams in which the image pairs have a fixed functional relationship that is learned by the network. Relational learning will be described in detail in a future document.

Chapter 4 discusses the results and presents conclusions.

This work is part of project 12ph - Autonomous Land Systems. Our long term goal is to demonstrate self-organized, adaptive learning in a simulated mobile vehicle with multiple sensors. The techniques would be applicable to an autonomous vehicle operating in an unknown and dynamically changing environment.

2 An Associative Memory RNN

We have previously reported associative memory (AM) in an RNN [4]. This chapter reviews the structure and learning algorithms of the AM RNN that is used in Chapter 3 to demonstrate the advantages of our new weight constraint technique.

Two memory regions in the AM RNN are used to store and regenerate patterns from two independent sensor arrays. These are termed the image array (S_I) and its associated code (S_C).

The image array is connected to a central region of the recurrently connected nodes (R nodes). Each pixel in the image has a fixed number of randomly chosen connections to R nodes in the central region. No two image pixels have the same set of output connections.

Output from the RNN is presented as two arrays that are connected to the separate R node memory regions. These memory node arrays, M_I and M_C, are intended to reproduce S_I and S_C respectively after the training process.

It is only input from S_I that generates responses in the memory arrays M_I and M_C, and during training there is feedback from the differences between corresponding pixels in the image and code arrays. This feedback adjusts the connection weights to R nodes in the memory regions so that the differences are reduced. The details are reviwed in this chapter.

The weight that connects two R nodes modifies the signal transferred between the nodes by a multiplicative factor. Each R node has the same number of input connections, established randomly after the input connections from the sensor nodes (S nodes) have been chosen. An R node can be connected to any other R node in the RNN, but self-connection is not permitted.

The weights are all positive in this RNN, and each R node has a fixed output sign. The nodes in the RNN are thus either excitatory (positive) or inhibitory (negative). The signs are chosen randomly, and the fraction of positive nodes (normally 0.5) is defined by the user.

The M_I and M_C nodes (M nodes in general) have input connections that are chosen randomly from R nodes in the memory regions, with the requirement that each M node has an equal number of inputs from excitatory and inhibitory R nodes.

The connection sets for all nodes are unique. No S node can have more than one output to any R node, and no two S nodes can have the same set of outputs. No M node can have more than one input from any R node, and no two M nodes can have the same set of inputs. No R node can have more than one input from any other node.

The weights for connections from S nodes to R nodes, and from R nodes to M nodes are kept constant, because these connections simply transfer information. Any learning occurs within the R nodes. The magnitudes of the R node connection weights are initially random,

between the limits supplied by the user; e.g. 0.1 to 0.9. The S to R node weights are all initially set to a user specified constant (default 1.0), as are the R to M node weights.

2.1 Node Response Functions

An image and its associated code are presented to the RNN through the sensor arrays. Each pair of patterns is held in the sensor arrays for a fixed number of RNN cycles. We call this the exposure time. During training, the weights connecting the R nodes are adjusted at each cycle, according to the learning rules given later in this chapter.

The nodes respond by passing a weighted sum of the incoming signals through a sigmoid function. The responses are calculated at each cycle of the RNN. The weighted input (x_n) to the nth node at cycle t is:

$$x_n^t = \sum_i^{n_c} W_{ni}^{t-1} \sigma_j f_j^{t-1} \tag{1}$$

where n_c is the number of input connections, and the subscript j gives an index for the node providing input on the ith connection, with weight W_{ni}^{t-1}. The input connection indices are stored in a matrix C, thus $j = C_{ni}$. The output of the node on the jth connection is f_j^{t-1}, and σ_j is its sign.

The sigmoid function has an exponent s, controlling the steepness of the function, and an offset x_0, giving the centre of its output range. The output of the nth node is then:

$$f_n^t = [1 + \exp(-s(x_n^t - x_{0n}))]^{-1} \tag{2}$$

Fixed values for the sigmoid exponents (s_r) are used for every R node. The M nodes use a separately defined fixed value (s_m). The sensor nodes simply supply values between 0.0 and 1.0.

Each R node offset is set at the value that centres its output at 0.5 when all the connected nodes provide a signal of 0.5. This value varies for each R node, but is kept constant after being calculated during the RNN initialization. We have found that attempting to optimize the offsets does not improve the RNN performance. The M node offsets are set to zero.

Output signals for the R nodes are initialized randomly between 0.0 and 1.0.

2.2 Changing the Connection Weights

During the RNN training process the connection weights are changed after each network cycle; i.e. after all the node outputs have been calculated for a fixed input array. An algorithm used to change the weights is called a learning rule.

For associative memory of image pairs we have used two learning rules. For R nodes that are not connected to memory arrays we used a form of Hebbian learning, which is based on a mechanism proposed by Hebb [9] for biological systems. For R nodes that provide input to the memory arrays (M_I and M_C), we used a difference feedback algorithm.

In our form of Hebbian learning, a connection weight is increased when strong input results in a strong output from the receiving node.

For an R node n at cycle t, which received an input f_j^{t-1} from the jth node and generated an output f_n^t, the change in the weight W_{ni} connecting nodes n and j is:

$$\Delta W_{ni}^t = \alpha f_j^{t-1} f_n^t - \gamma W_{ni}^t \tag{3}$$

where α and γ control the growth and decay rates respectively. Recall that the connection index j is given by C_{ni}.

Note that the weights are constrained by the decay term in Equation 3.

Our difference feedback learning adjusts the input connection weights of each R node that provides output to one or more M nodes, so that the correlation between each M and S array can be maximized.

For the nth R node that is connected to one or more M nodes, we first calculate a sum of differences between the signals in each connected M node and its corresponding S node, in the following way:

$$D_n = \frac{\sigma_n}{n_{cm}} \sum_i^{n_{cm}} \Delta_i \tag{4}$$

where Δ_i is the difference between the outputs generated by the ith S and M nodes, n_{cm} is the number of connections to M nodes from the nth R node, and σ_n is the output sign of the R node.

M node arrays (M_I,M_C) for the image and its code are connected to different regions of the RNN, so the summation only involves image differences or code differences for a given R node.

D_n defines the required direction for the change in output signal strength of the nth R node. Dividing by n_{cm} confines the magnitude of D_n to [0.0, 1.0]; i.e. it is a fractional value.

For the nth R node, the change in the jth input weight at cycle t is calculated in the following way:

$$\Delta W_{jn}^t = \rho D_n^t f_n^t \sigma_k f_k^t W_{jn}^t \tag{5}$$

where ρ is the feedback learning rate. The magnitude of a weight change depends on the strength of the incoming signal f on the jth input connection (from the R node whose index $k = C_{jn}$), and on the strength of the weight itself (W_{jn}).

With our form of difference feedback learning, the weight changes become smaller as the S and M arrays become correlated (D_n^t approaches zero). On the other hand, there is no explicit constraint on the weights affected by this learning algorithm and if correlation is not achieved smoothly it may lead to instability. The sigmoidal weight constraint technique described in Section 2.4 removes this problem.

2.3 The Correlation Function

To assess the effectiveness of an RNN in recalling a set of patterns, we used the following correlation function:

$$C = 1 - \frac{1}{Nn_s} \sum_{j}^{N} \sum_{i}^{n_s} |\Delta_i|_j \tag{6}$$

$|\Delta_i|_j$ is the difference between the ith S and M nodes for the jth pattern, N is the number of patterns, and n_s is the number of sensor pixels. If every pattern is accurately reproduced during training, C approaches 1.0 ($|\Delta_i| \rightarrow 0$). During training, pattern j is held in the S node array for the exposure time, e_t. After e_t cycles, the i sum is calculated and the next pattern is presented for e_t cycles. C is calculated over each complete cycle of all patterns. Since there are two sensor arrays, there are actually two correlations calculated, C_i and C_c, for the image and code arrays respectively.

2.4 Sigmoidal Weight Constraint

Consider the sigmoid function given by:

$$y(X) = \frac{1}{[1 + \exp(-4.0(X - 0.5))]} \tag{7}$$

This has, of course, the same functional form as that used to generate R node outputs (Equation 2), but with fixed values for the exponent (4.0) and offset (0.5). Note that we use a capital X for the independent variable of Equation 7 to differentiate it from the node inputs of Equations 1 and 2. This sigmoid function is shown in Figure 1.

If we consider y to represent a connection weight, then we may define a related variable X from the inverse of Equation 6, thus:

$$X = 0.5 - [log((1.0/y) - 1.0)/4.0] \tag{8}$$

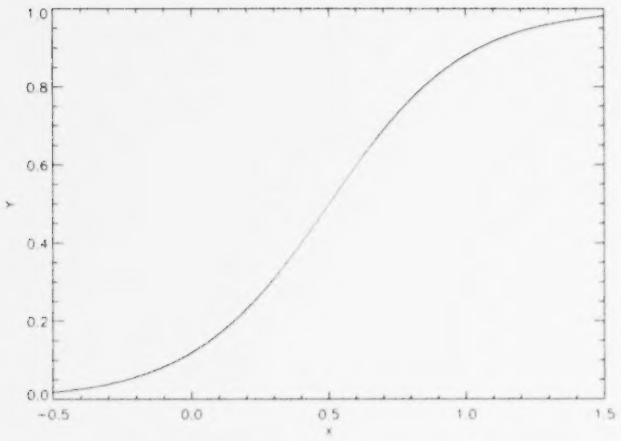

Figure 1: *Sigmoidal curve used to constrain weights*

So for any connection weight we define the related X, and during training we **change X in place of changing the weight**.

Our Hebbian learning then becomes (cf. Equation 3):

$$\Delta X_{ni}^{t} = \rho(f_j^{t-1} f_n^t - 0.25) \tag{9}$$

and our difference feedback becomes (cf. Equation 5):

$$\Delta X_{jn}^{t} = \rho D_n^t f_n^t \sigma_k f_k^t W_{jn}^t \tag{10}$$

Note that now there is no parametric decay term in the Hebbian algorithm, and a single, common learning rate parameter ρ is used in both algorithms. In the new Hebbian algorithm, if both sending and receiving nodes are at the middle of their output (0.5), then the weight is unchanged (the product is 0.25).

Whenever the actual weight value is required in the RNN calculation cycles, it is obtained from Equation 7, using the current value of X. In this way, **the weights are confined to move smoothly on the sigmoidal curve of Figure 1, and the weight values are constrained to the range [0.0, 1.0]**.

Because we have used 4.0 for the exponent and 0.5 for the offset in Equation 7, there is an approximately linear relationship between X and the weight near the centre of the weight range (0.5); i.e. near 0.5, changing X is essentially the same as changing the weight.

It is intuitively clear that in any biological neural system there must be some physical limits for each connection strength. These physical limits can be represented by 0.0 and 1.0. Of course, in a computer system the double precision values give a massive range for the scale of relative strengths.

3 Results

3.1 Long Term Instability with No Sigmoidal Weight Constraint

Previously [4] we presented results showing the convergence of the correlations for sets of images and their associated codes. A set of ten 4x5 patterns and 4x1 code vectors were used as a test problem.

We showed that a pattern and its identification code can be stored by the simultaneous presentation of two image streams during training, and an identification can subsequently be recovered from the code memory array that is generated by the single presentation of a pattern in the image sensor stream.

The training process is affected by the following parameters: the constant value used for the S node to R node weights; the learning parameters, α, β, γ and ρ; the node sigmoid steepness factors s_r, and s_m; the number of R nodes; the number of output connections for the S nodes; the number of input connections for the R and M nodes; the exposure time, e_t; and the fraction of positive (excitatory) R nodes.

For a given RNN structure containing 500 R nodes, the fraction of positive R nodes was fixed at 0.5, the number of R node input connections was 40 and the number of output connections for each S node was also 40. The image input and memory regions had 200 nodes each, and the code memory region had 100 nodes. The optimal number of input connections for the memory node arrays (M_i and M_c) was 40. Optimal values were then established for the learning parameters [4].

It was found that if the learning rates (α, ρ) are too high the training could become unstable, but that apparent stability could be maintained by keeping these values small enough; e.g. $\rho = 0.004$ maintained stability when training 2 image pairs, but $\rho = 0.001$ was required when ten pairs were used.

It appeared that stable convergence had been achieved for 10 pairs after 6,000 training cycles, but as Figure 2 shows, if the training process is allowed to continue (20,000 cycles are shown) the process becomes erratic.

With $\rho = 0.001$ it takes about 6,000 training cycles to obtain correlations greater than 0.86. If $\rho = 0.002$ is used, high corelations are attained after about 2,000 cycles, but the long term behaviour becomes even more unstable (chaotic), as seen in Figure 3.

3.2 Stable, Rapid Convergence with Sigmoidal Weight Constraint

When the sigmoidal weight constraint technique is introduced, the performance of the RNN is dramatically improved. Figure 4 shows the long term correlation for 10 image pairs using $\rho = 0.001$. This can be directly compared to the curve of Figure 2 as they used the

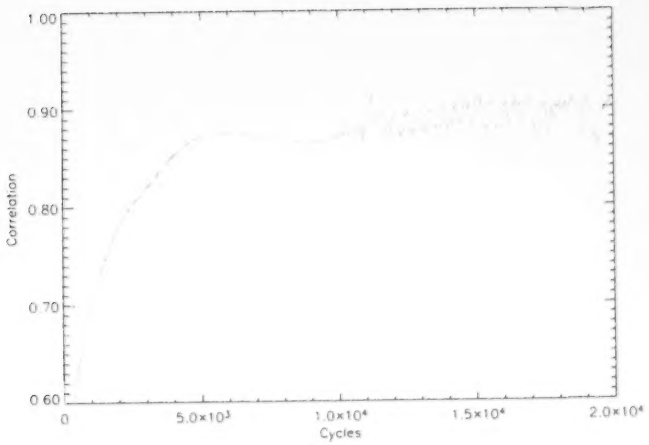

Figure 2: *Erratic correlations for images (lower) and codes (upper), using 500 R nodes to train 10 images* ($\rho = 0.001$)

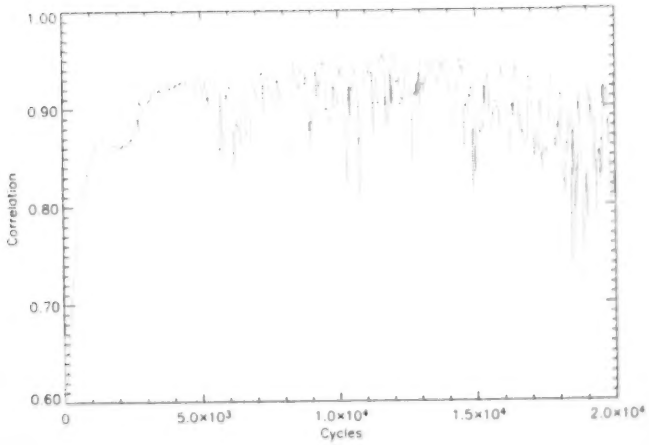

Figure 3: *Unstable correlations for images (lower) and codes (upper), using 500 R nodes to train 10 images* ($\rho = 0.002$)

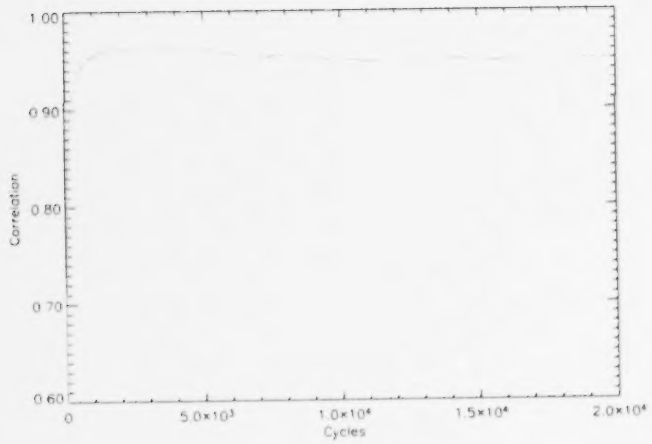

Figure 4: *Correlations for 10 images (lower) and codes (upper), using Sigmoidal Weight Constraint ($\rho = 0.001$)*

same RNN configuration. Apart from the evident long term stability, Figure 4 shows faster convergence and a greater maximum correlation.

Figure 5 shows the long term correlation for 10 image pairs using $\rho = 0.002$, and can be compared with the curve of Figure 3. In this case, increasing the learning rate parameter ρ does not lead to instability, and does give faster convergence to an increased maximum correlation.

In both cases, with sigmoidal weight constraint the RNN clearly remains stable indefinitely. Furthermore, the maximum correlations for code and image sets occur after about 500 cycles rather than about 5,000 cycles as in Figures 2 and 3. Finally, the accuracy of the maximum correlations is greater than 0.96 for both image and code sets when sigmoidal constraint is used. Without it a stable maximum of only about 0.86 was possible for the images (Figure 2).

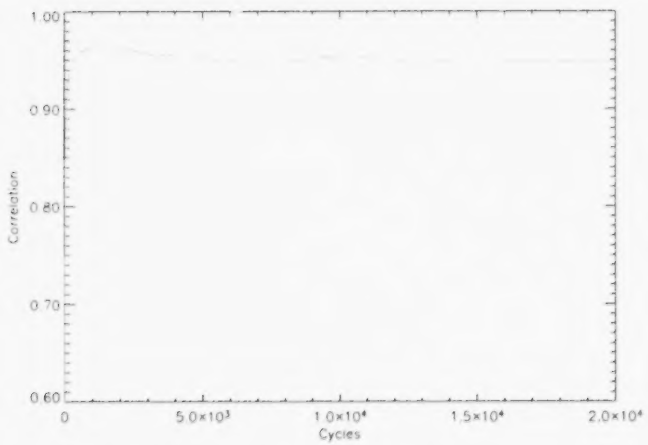

Figure 5: *Correlations for 10 images (lower) and codes (upper), using Sigmoidal Weight Constraint* ($\rho = 0.002$)

4 Discussion and Conclusions

Using RNNs that are structured to generate associative memory, we have shown that numerical instability can arise while the network connection weights are being altered during the training process. By constraining the weights to move smoothly along a sigmoidal curve the instability is removed.

Sigmoidal weight constraint also has the following additional advantages:

- weight magnitudes are controlled without the need for renormalization or the use of a parametrized decay term;
- the number of network cycles required to reach the maximum pattern correlation is reduced by a factor of about ten; and
- the accuracy of the pattern correlations is significantly improved.

Computational time is thus greatly reduced and better accuracy is achieved for any RNN application that uses a trained network. Continuous training may also be employed while maintaining numerical stability.

In the associative memory application, a pattern and its identification code can be stored by the simultaneous presentation of two sensor image streams during training. After training, the correlations between input and output node arrays are high for a previously seen pattern, and an image may then be directly identified by the output in the code memory array.

We have also used sigmoidal weight constraint in an RNN structure that is capable of relational memory. In this case sets of two input patterns have a fixed relationship that is learned through the training process, and the relationship is provided as an output array from the network. Here the use of sigmoidal weight constraint also greatly improves the network performance. There are many potential applications of relational learning, including the generation of depth maps from stereo image pairs, and the generation of control signals for a mobile machine that uses two or more types of sensor. The relational learning capability of RNNs will be the subject of future work.

The sigmoidal technique has also been employed in an RNN controlling the movement of a simulated mobile machine, again with smooth, stable performance. Combined with relational learning, this system will now be used to relate the input from two sensor arrays with the actions taken by a mobile machine in response to those sensors. The machine responses can be supplied either by a human user (driving the machine) or by an AI technique such as reinforcemnt learning. It should be possible to generate adaptive control in the mobile machine; i.e the ability to navigate through obstacles and reach a goal would improve with experience. The use of continuous learning in such an RNN-based system benefits greatly from the stability provided by the sigmoidal weight constraint technique.

References

1. Barton, S.A., "Structure and Convergence Properties of a Recurrent Neural Network", DRES SM-1489, 1996.

2. Barton, S.A., "Techniques for Pattern Classification Using a Convergent Recurrent Neural Network", DRES SR-709, 1998.

3. Barton, S.A., "Recognition and Identification of Objects in IR Feature Images using a Recurrent Neural Network", CAN contribution to TTCP W7 KTA 7-2, Final Report, 2000.

4. Barton, S.A., "Associative Memory in a Recurrent Neural Network", DRES TM 2001-053, 2001.

5. Barton, S.A., "The Effect of Sensory Input on Trajectories Generated by Recurrent Neural Networks", DRES SR-589, 1993.

6. Barton, S.A., "Two Dimensional Movement Controlled by a Chaotic Neural Network", Automatica, **31** (1995), p. 1149.

7. Kohonen, T, *Self-Organization and Associative Memory*, Springer, Berlin, 1984.

8. Miller, K.D. and MacKay, D.J.C., "The Role of Constraints in Hebbian learning", Neural Comput., **6**, (1994), p. 100.

9. Hebb, D.O., *The Organization of Behaviour*, John Wiley and Sons, Inc., New York, 1949.

UNCLASSIFIED

SECURITY CLASSIFICATION OF FORM

(highest classification of Title, Abstract, Keywords)

DOCUMENT CONTROL DATA

(Security classification of title, body of abstract and indexing annotation must be entered when the overall document is classified)

1. ORIGINATOR (the name and address of the organization preparing the document. Organizations for who the document was prepared, e.g. Establishment sponsoring a contractor's report, or tasking agency, are entered in Section 8.)

   Defence R&D Canada – Suffield
   PO Box 4000, Station Main
   Medicine Hat, AB T1A 8K6

2. SECURITY CLASSIFICATION (overall security classification of the document, including special warning terms if applicable)

   Unclassified

3. TITLE (the complete document title as indicated on the title page. Its classification should be indicated by the appropriate abbreviation (S, C or U) in parentheses after the title).

   Sigmoidal Weight Constraint in a Recurrent Neural Network (U)

4. AUTHORS (Last name, first name, middle initial. If military, show rank, e.g. Doe, Maj. John E.)

   Barton, Simon A.

5. DATE OF PUBLICATION (month and year of publication of document)

   December 2004

6a. NO. OF PAGES (total containing information, include Annexes, Appendices, etc) 24

6b. NO. OF REFS (total cited in document) 9

7. DESCRIPTIVE NOTES (the category of the document, e.g. technical report, technical note or memorandum. If appropriate, enter the type of report, e.g. interim, progress, summary, annual or final. Give the inclusive dates when a specific reporting period is covered.)

   Technical Memorandum

8. SPONSORING ACTIVITY (the name of the department project office or laboratory sponsoring the research and development. Include the address.)

9a. PROJECT OR GRANT NO. (If appropriate, the applicable research and development project or grant number under which the document was written. Please specify whether project or grant.)

9b. CONTRACT NO. (If appropriate, the applicable number under which the document was written.)

10a. ORIGINATOR'S DOCUMENT NUMBER (the official document number by which the document is identified by the originating activity. This number must be unique to this document.)

   DRDC Suffield TM 2004-261

10b. OTHER DOCUMENT NOs. (Any other numbers which may be assigned this document either by the originator or by the sponsor.)

11. DOCUMENT AVAILABILITY (any limitations on further dissemination of the document, other than those imposed by security classification)

   (x) Unlimited distribution
   ( ) Distribution limited to defence departments and defence contractors; further distribution only as approved
   ( ) Distribution limited to defence departments and Canadian defence contractors; further distribution only as approved
   ( ) Distribution limited to government departments and agencies; further distribution only as approved
   ( ) Distribution limited to defence departments; further distribution only as approved
   ( ) Other (please specify):

12. DOCUMENT ANNOUNCEMENT (any limitation to the bibliographic announcement of this document. This will normally corresponded to the Document Availability (11). However, where further distribution (beyond the audience specified in 11) is possible, a wider announcement audience may be selected).

   Unlimited

13. ABSTRACT (a brief and factual summary of the document. It may also appear elsewhere in the body of the document itself. It is highly desirable that the abstract of classified documents be unclassified. Each paragraph of the abstract shall begin with an indication of the security classification of the information in the paragraph (unless the document itself is unclassified) represented as (S), (C) or (U). It is not necessary to include here abstracts in both official languages unless the text is bilingual).

When training a recurrently connected neural network (RNN), the magnitude of the connection strengths (weights) must be limited in some way. The weights are normally constrained by either renormalizing them after each learning step, or by using a decay term proportional to the weight. For large numbers of training cycles, we show that an RNN output can become unstable with previously used weight adjustment methods.

We introduce a technique that constrains weight values to move on a smooth sigmoidal curve. Without the need for renormalization or a parametric decay term, our RNNs then produce stable output. Performance is also improved in other ways. As an example, an associative memory RNN is shown to converge much faster and to more accurate values than with previous methods.

14. KEYWORDS, DESCRIPTORS or IDENTIFIERS (technically meaningful terms or short phrases that characterize a document and could be helpful in cataloguing the document. They should be selected so that no security classification is required. Identifies, such as equipment model designation, trade name, military project code name, geographic location may also be included. If possible keywords should be selected from a published thesaurus, e.g. Thesaurus of Engineering and Scientific Terms (TEST) and that thesaurus-identified. If it is not possible to select indexing terms which are Unclassified, the classification of each should be indicated as with the title.)

Autonomous vehicles; Artificial intelligence; Neural Networks; Pattern recognition